(12) UK Patent Application (19) GB (11) 2 371 889 (13) A

(43) Date of A Publication 07.08.2002

(21) Application No 0102706.9

(22) Date of Filing 02.02.2001

(51) INT CL[7]
G06F 17/30

(52) UK CL (Edition T)
G4A AUDB

(56) Documents Cited
WO 1998/018246 A2
SMPTE JOURNAL, VOL.109, JULY 2000, J.WILKINSON, "THE SMPTE DATA CODING PROTOCOL & DICTIONARIES", PP.579-586, SEE ALSO INSPEC ABSTRACT ACCESSION NO. 6721141
HTTP://WWW.AAFASSOCIATION.ORG/HTML/TECHINFO/AAF_DEV_OVERVIEW .PDF "AAF AN INDUSTRY-DRIVEN OPEN STANDARD FOR MULTIMEDIA AUTHORING" (11 APRIL 2000)

(58) Field of Search
UK CL (Edition S) G4A AUDB
INT CL[7] G06F 17/30 17/60
ONLINE: EPODOC, WPI, PAJ, INSPEC, INTERNET

(71) Applicant(s)
Sony United Kingdom Limited
(Incorporated in the United Kingdom)
The Heights, Brooklands, WEYBRIDGE, Surrey, KT13 0XW, United Kingdom

(72) Inventor(s)
James Hedley Wilkinson
Alan Turner

(74) Agent and/or Address for Service
D Young & Co
21 New Fetter Lane, LONDON, EC4A 1DA, United Kingdom

(54) Abstract Title
Data structures

(57) A data structure comprises KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field. The KLV packets are organised into: a File Header including a data space for metadata associated with data in a File Body; at least one of a first KLV packet forming a first component of a File Body and a second KLV packet forming a second component of the File Body; and a File Footer.

The first component , if present, has a Value field containing location data directly or indirectly indicating the location of first essence which is not included in the File Body, and a Key field the Value of which indicates that the Value field of the first component contains the said location data. The second component, if present, having a Value field containing an essence container for containing second essence, and a Key field the Value of which identifies the second essence. One or both of the first and second essence is associated with the said metadata.

The Key field of the first components has the same form as the Key field of the second component, the components being distinguished by distinguishing codes in the Key fields.

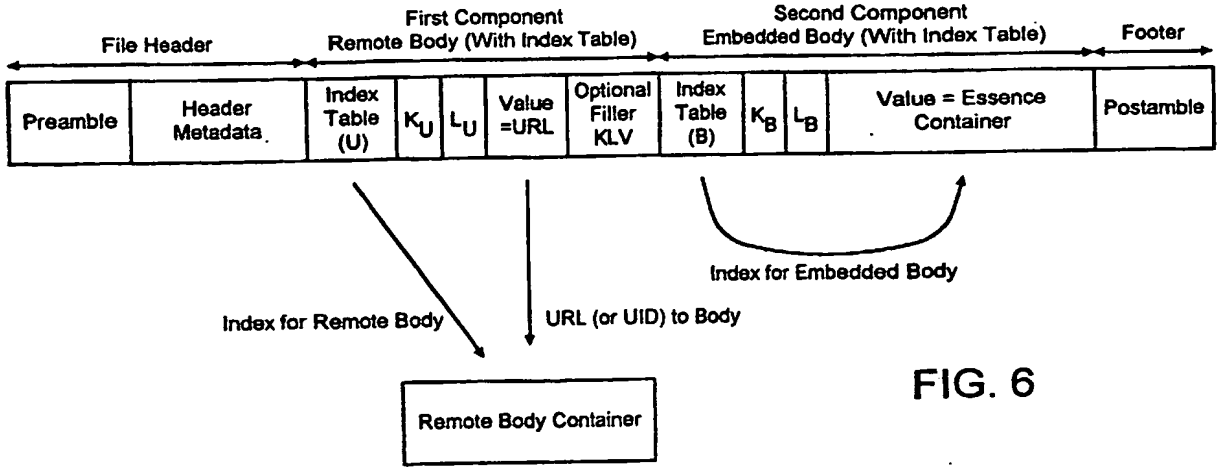

FIG. 6

GB 2 371 889 A

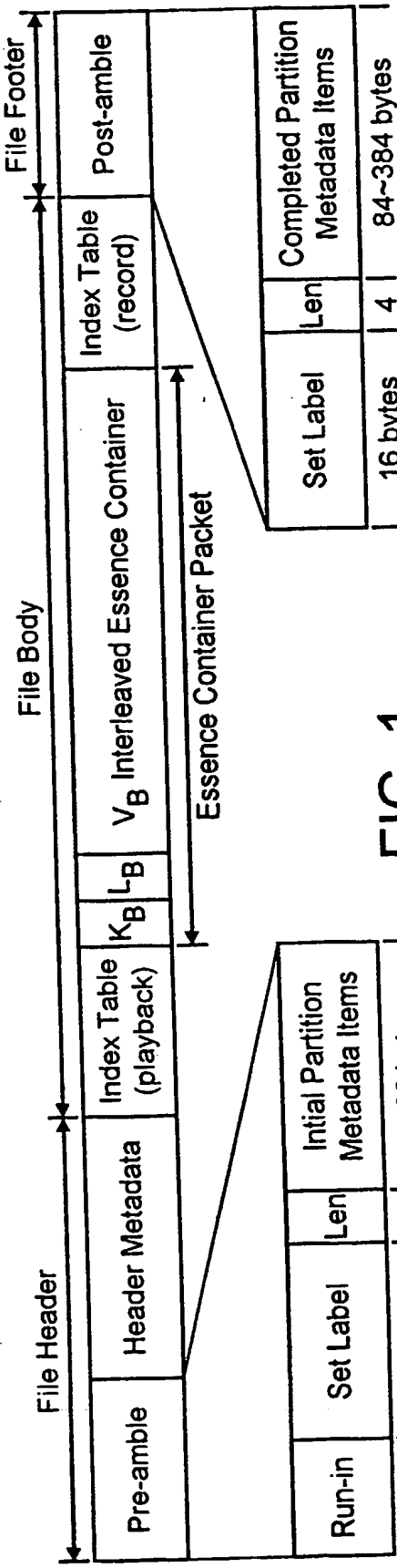

FIG. 1

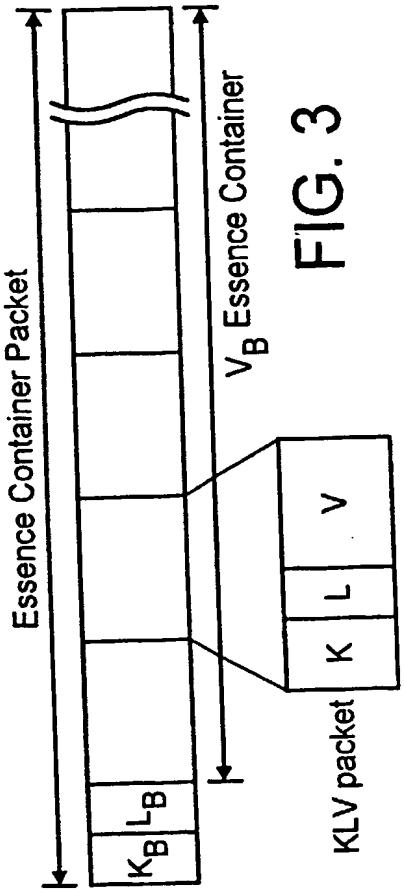

FIG. 3

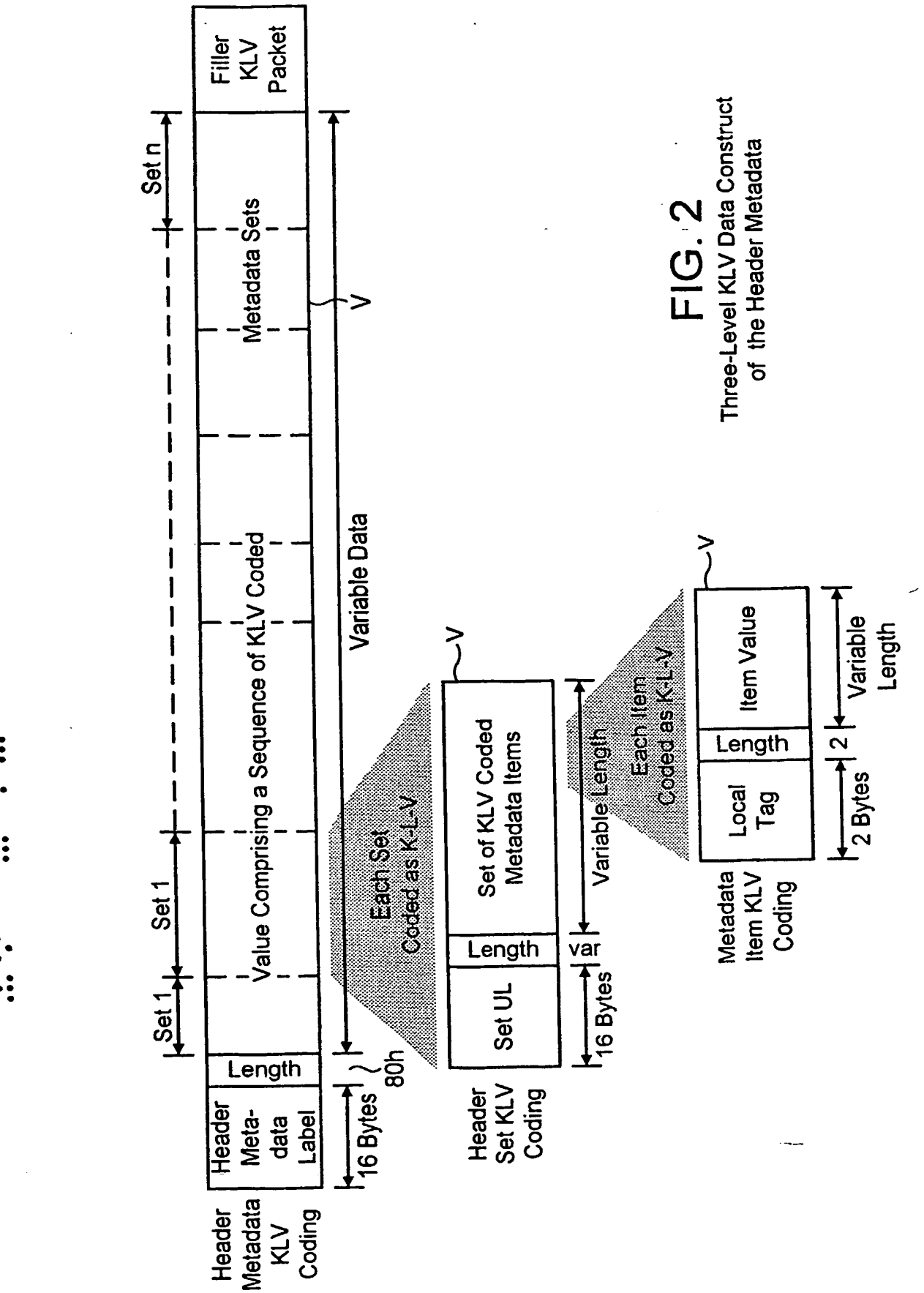

FIG. 2
Three-Level KLV Data Construct of the Header Metadata

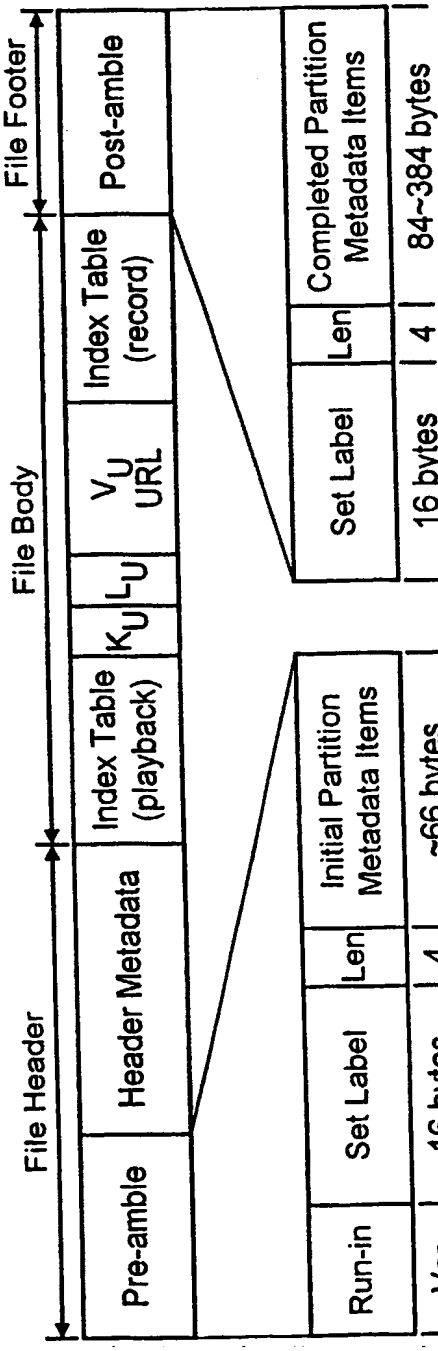

FIG. 4

MXF file with linked body

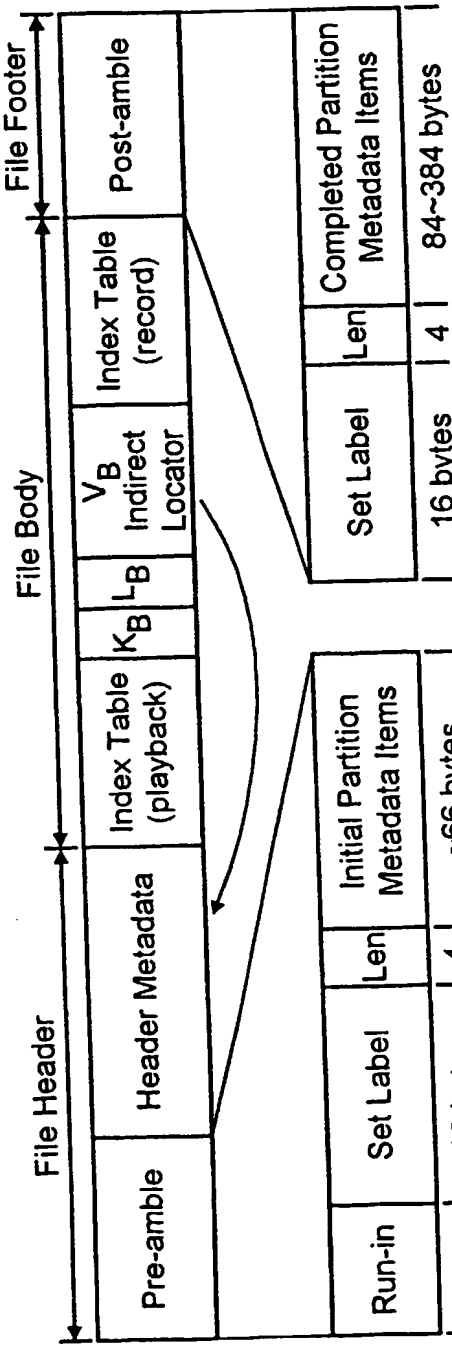

FIG. 5

MXF file with indirectly linked body

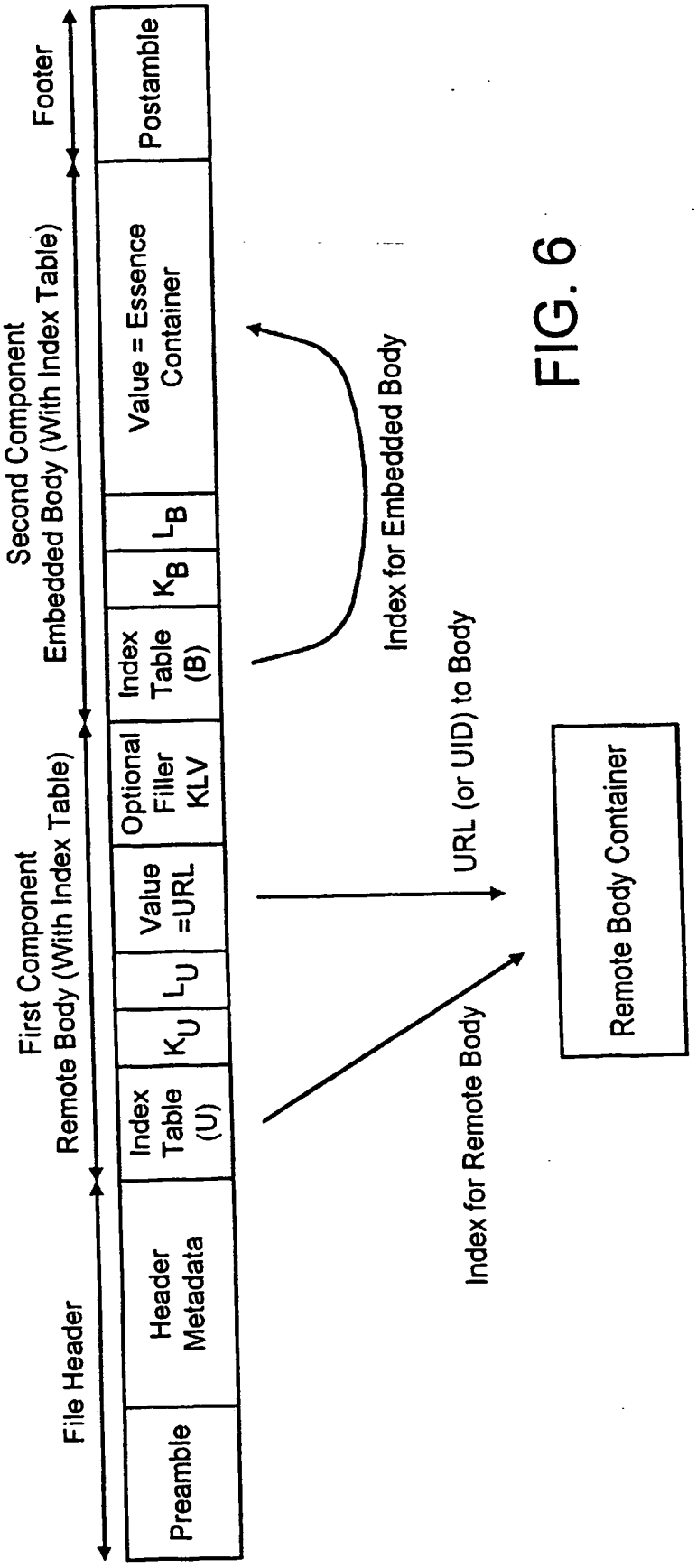

FIG. 6

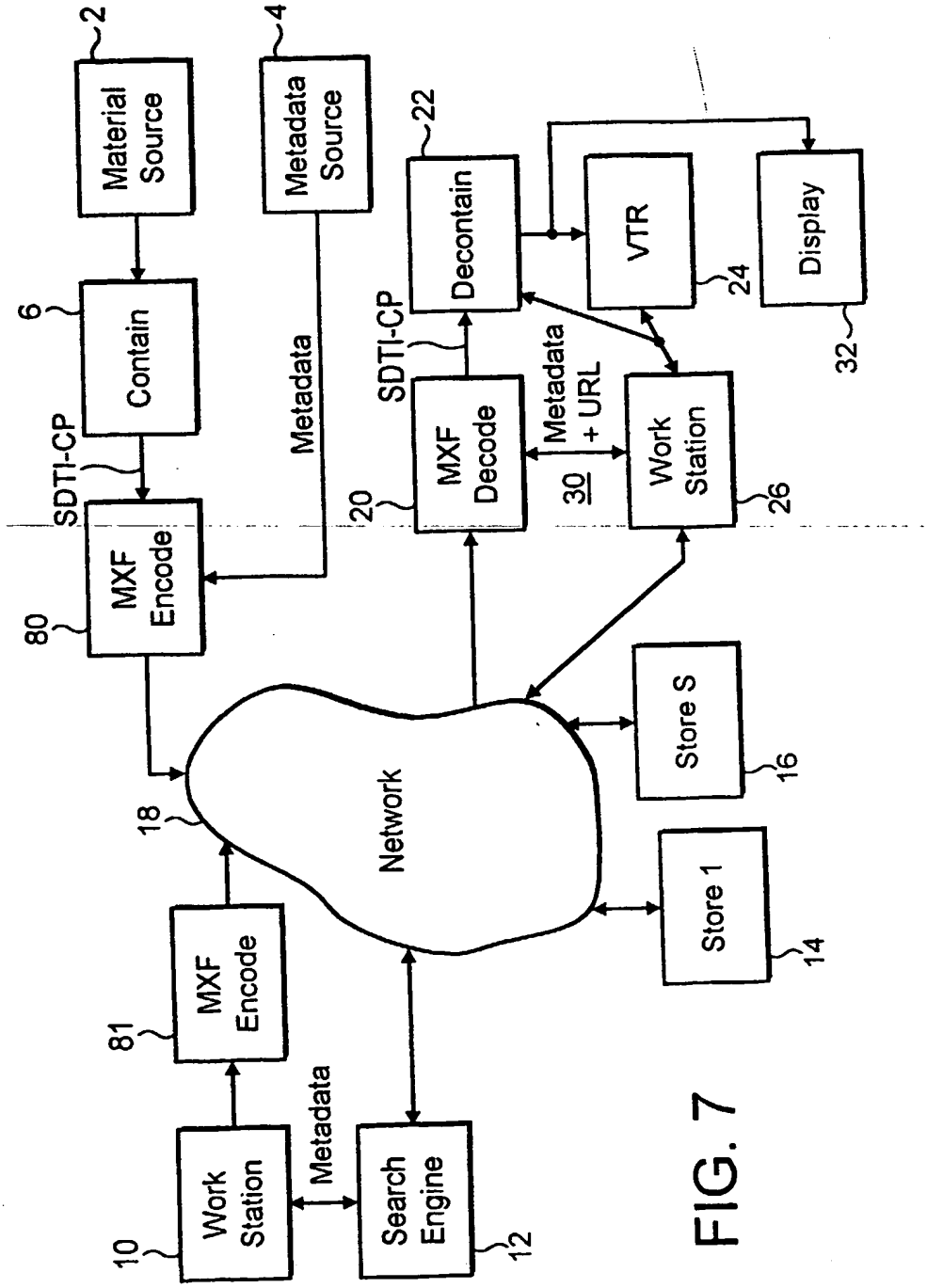

FIG. 7

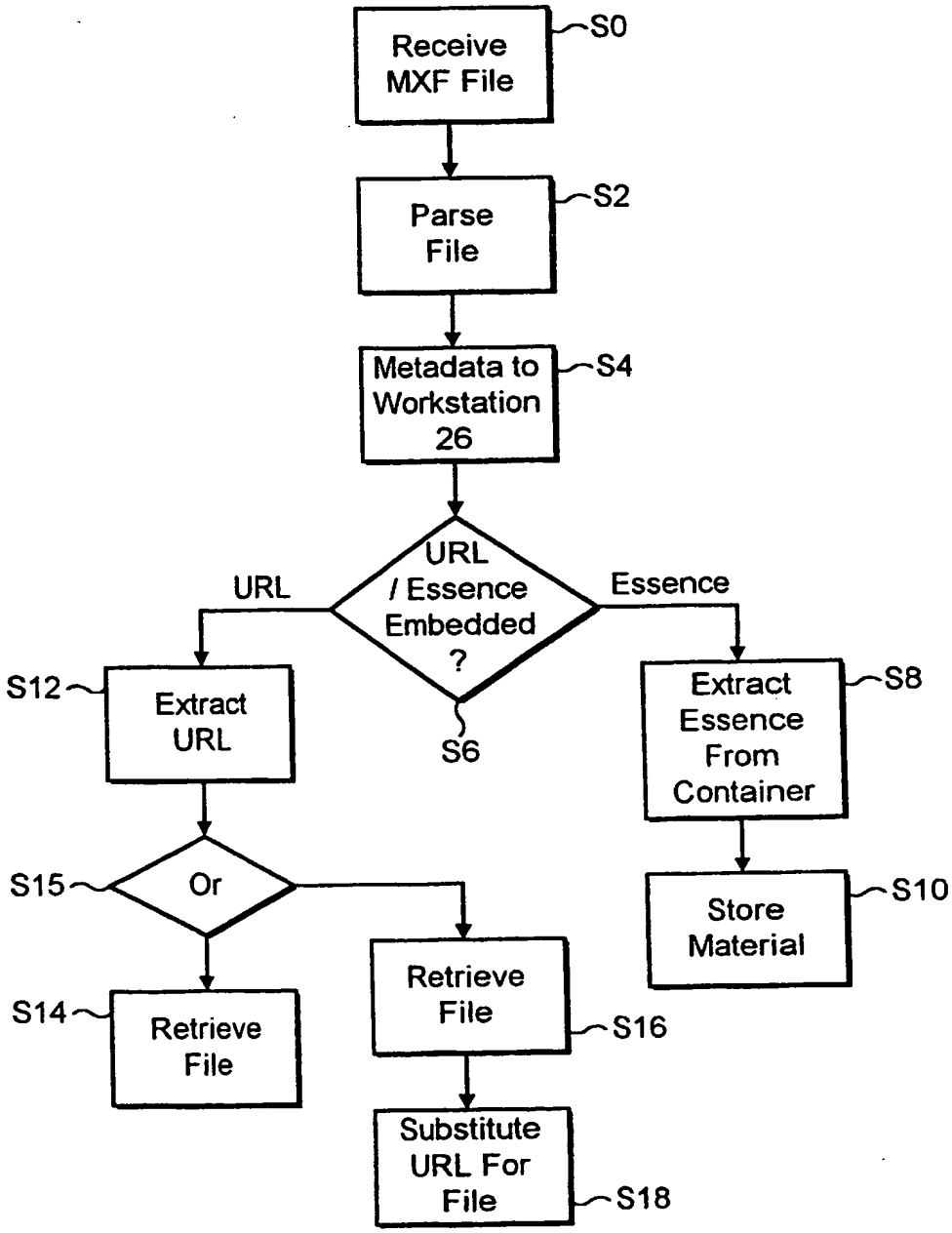

FIG. 8

Data Structures

The present invention relates to a data structure, a method of, and apparatus for, forming a data structure, and a computer program product. Preferred embodiments of the invention relate to MXF (Material eXchange Format) Files and the invention and its background will be described in relation to MXF Files.

MXF is described in the conference publication of the International Broadcasting Convention IBC 2000, pages 395 to 400.

The purpose of MXF Files is to allow programme material together with Metadata associated with the material to be transferred between material processing and storage equipment. Material is one or more of video essence, audio essence and data essence together with essential formatting metadata embedded in a suitable container. Metadata is any information relating to, or associated with, the material. Essence is the video, audio and data content. For example, video essence is data representing only the raw sampled video.

MXF Files as currently proposed include a File Header which includes the metadata in a section termed Header Metadata, followed by a File Body containing the material followed by a File Footer. Thus a complete File of the material together with its associated metadata is provided for transfer. The Header, Body and Footer are coded using a sequence of Key, Length, Value (KLV) Packets. KLV coding is described in the SMPTE Journal July 2000, Vol. 109, No 7, Engineering Report.

However, some storage systems store the material separately from the associated metadata. Commonly library and archive systems store the material separately from the associated metadata. The metadata is made available to a search tool which then locates the associated material without having to spool through possibly many hours of material.

According to a first aspect of the present invention, there is provided a data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with data in a File Body;

at least one of a first KLV packet forming a first component of a File Body and a second KLV packet forming a second component of the File Body;

the first component , if present, having a Value field containing location data directly or indirectly indicating the location of first essence which is not included in the File Body, and a Key field the Value of which indicates that the Value field of the first component contains the said location data;

the second component, if present, having a Value field containing an essence container for containing second essence, and a Key field the Value of which identifies the second essence;

one or both of the first and second essence being associated with the said metadata; and

a File Footer;

wherein the Key field of the first components has the same form as the Key field of the second component, the components being distinguished by distinguishing codes in the Key fields.

Thus the Key field has the same, consistent form for both a component having embedded essence and a component having a locator indicating the location of essence which is not embedded in the data structure. This simplifies the encoding and decoding of data structures when transferring them in networked systems.

The first essence may be full resolution essence and the second essence may be a low resolution version of the first essence. That allows the File structure to be used to browse the low resolution essence without needing to access the full resolution essence. Alternatively, the first essence may be the low resolution version and the second essence may be the full resolution essence.

Preferably the first component of the data structure includes a filler KLV packet for containing filler data so that the first component has a predetermined length.

Preferably the said file header includes a filler KLV packet for containing filler data so that the file header has a predetermined length.

A first embodiment of the first aspect provides a data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes

the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with data in a File Body;

a KLV packet forming at least part of the File Body and having a Value field containing an essence container, a Key field including a predetermined code indicating the the Value field contains an essence container and , a Length field containing a predetermined code independent of the amount of data in the Value field; and

a File Footer.

A second embodiment of the first aspect provides a data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with essence;

a KLV packet forming at least part of the File Body having a Key field, a Value field and a Length field indicating the amount of data in the Value field; and

a File Footer;

wherein the Value field of the said a KLV packet forming at least part of the File Body contains location data directly or indirectly indicating the location of the essence container which essence is associated with the said metadata and which is not contained in the Value field of the File Body and the Key field of the said KLV packet forming at least part of the File Body has a predetermined code indicating that the Value field .contains the said location data.

In one preferred version of the second embodiment, the Key field of the File Body indicates that the Value field contains location data.

In another preferred version of the second embodiment, the said location data in the Value field is a pointer which indicates that a locator indicating the location of the essence container is in the said data space for metadata.

In yet another preferred embodiment, the said location data in the Value field is a unique identifier which is used by a resource locator device to provide the location of the essence container.

Thus the invention provides a consistent data structure for Files which have embedded essence and Files which do not have embedded essence but which instead have location data indicating the location of an essence container.

The invention allows the use of a simple parser to determine the high level structure of the File and to simply detect the metadata container and either the essence container) or location data relating to the essence container.

In the said second embodiment of the invention, the locator allows the material associated with the said Metadata to be located and retrieved from a data store by a system which receives the data structure. The absence of the File Body reduces the size of the File and reduces the time needed to transfer the data structure.

In a version of said second embodiment, in which the said location data in the Value field is a pointer which indicates that a locator indicating the location of the essence container is in the said data space for metadata, the pointer preferably indicates the location in the data space for metadata of the locator. The data space for metadata can have a complex multi-layer structure of KLV packets. The pointer simplifies the task of finding the location data.

The invention also provides a signal comprising a file structure according to the first aspect of the invention.

A second aspect of the invention provides a method of creating a File structure, comprising the steps of:

receiving digital data representing essence;

receiving metadata relating to the said essence; and

receiving location data relating to the location of further essence; and

forming a data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, ;

the method comprising organising the said KLV into:

a File Header including a data space for metadata associated with data in a File Body;

at least one of a first KLV packet forming a first component of a File Body and a second KLV packet forming a second component of the File Body;

the first component if present, having a Value field containing location data directly or indirectly indicating the location of first essence which is not included in the File Body, and a Key field the Value of which indicates that the Value field of the first component contains the said location data;

the second component, if present, having a Value field containing an essence container for containing second essence, and a Key field the Value of which identifies the second essence;

one or both of the first and second essence being associated with the said metadata; and

a File Footer;

wherein the Key field of the first components has the same form as the Key field of the second component, the components being distinguished by distinguishing codes in the Key fields.

A third aspect provides apparatus arranged to create a File structure as defined in the first aspect of the invention.

Still another aspect of the invention provides a system for decoding a File structure, the system being arranged to::

receive a File structure which may be as defined in the said first aspect;

parse the structure;

determine from a KLV whether the Value field of the KLV packet contains the essence container or contains location data indicating the location of essence container; and

if the said Value field contains the essence container extract the essence container from the Value field and if the Value field contains the said location data extract the location data therefrom.

Still another aspect of the invention provides a computer program product arranged to carry out the method of said second aspect of the invention.

For a better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

Figure 1 is a schematic diagram of the overall data structure of a first example of an MXF File in accordance with the invention;

Figure 2 is a schematic diagram showing the data construct of Header metadata of Figure 1;

Figure 3 is a schematic diagram showing the data construct of File Body;

Figure 4 is a schematic diagram of the overall data structure of a second example of an MXF File in accordance with the invention;

Figure 5 is a schematic diagram of the overall data structure of a third example of an MXF File in accordance with the invention;

Figure 6 is a schematic diagram of the overall data structure of a fourth example of an MXF File in accordance with the invention;

Figure 7 is a schematic block diagram of an illustrative data processing system incorporating an example of the invention; and

Figure 8 is an flow chart illustrating a mode of operation of a material receiving system of the system of Figure 5.

First Example of an MXF File, Figure 1

Referring to Figure 1, there is shown the overall data structure of a File. Such a File is referred to herein as an MXF File (Material eXchange Format). The purpose of the Material Exchange Format is to exchange programme material together with attached metadata information about the material Body. The MXF File is intended for the transfer of programme material and metadata between disc-based File servers for example.

MXF Files are defined by a sequence of Key, Length, Value (KLV) coded data packets.

The first example of the File comprises a **File Header**, a **File Body** and a **File Footer**. In accordance with the first example of the File, the **Body** is **KLV** coded having a Key field K_B , a Length field L_B and a Value field V_B which in the example of Figure 1 is an interleaved essence container.

The Value field V_B or interleaved essence container contains the "essence" that is, in this example, digital video and/or audio essence. The essence may alternatively be, or additionally include, digital essence data. The following description refers only to digital video and audio for convenience of description.

It will be appreciated that Figure 1 is not drawn to scale. The Body is much greater than the preamble and post amble. The Body contains typically 99% or more of the total data content of the File.

The File Header contains the metadata associated with the essence in the essence container of the Body.

The File Header, the KLV coded File Body and the File Footer will now be described in more detail.

The File Header

The File Header contains a **Preamble** followed by **Header Metadata**. and optionally an **Index Table**.

The Pre-amble optionally starts with a Run-in byte sequence of 0 to 127 bytes e.g. 8 bytes. That is followed by a Key, Length Value (KLV) encoded partition metadata set which comprises:

an SMPTE Set Label of 16 bytes (the Key);

followed by one or more Length bytes, (4 in this example); which

is followed in this example by a Value field containing approximately 66 bytes of metadata for storage parameters used in the File. The Length byte indicates the amount of data in the Value field.

The Set Label defines the File as an MXF File.

Header Metadata, Figure 2.

The File Header includes a Header Metadata set.

Header Metadata set is structured as follows:

Individual metadata items coded as KLV,

Logical groupings of metadata items which are KLV coded metadata sets where the Value field of a set is a collection of KLV coded metadata items, and

where the top level, as shown in Figure 2, is the Header Metadata at the outermost level.

The foregoing is a simplified description. In practice, the actual implementation is defined by a Header Metadata Object Model which defines many levels where sets are logically connected to other sets through the application of unique identifiers.

The Header metadata may be any information associated with the essence of any kind contained in the File Body. The metadata may be descriptive of the essence, be technical data relating to the essence or any other information.

By way of example only, descriptive metadata may comprise data relating to the production of video material such as Programme ID, Title, Working Title, Genre ID, Synopsis, Director, Picturestamp, Keywords, Script, People, e.g. names and other details of performers and production crew.

By way of example, technical metadata may comprise data such as display aspect ratio, picture dimensions in pixels, picture rate, camera type, lens identification, and any other technical data.

Metadata may also comprise data relating to edits in the material. It may comprise instructions defining simple editing and other processes to be performed on the material.

Referring to Figure 2, the Header Metadata of the preamble comprises a 16 byte Header Metadata Label, followed by a Length field having a pre-determined code, in this example 80h, followed by KLV encoded metadata sets (sets 1 to *n*) which constitute the data of the Value field (V). The Value field is completed by a KLV "Terminating Filler" item which is interpreted as providing padding of the Header Metadata to a predetermined point in the File and providing a termination of the Header Metadata. The aim is for the sum of the variable Length Header Metadata sets and the Terminating Filler to be a constant. Preferably the filler item finishes at a convenient storage boundary.

The label of the Header defines the data Value (V) following the label as MXF Header Metadata.

Each KLV encoded metadata set *n* comprises a set label UL of 16 bytes, which uniquely identifies that set, one or more Length bytes and a set of KLV coded

metadata items constituting the data in the Value field V of the set. The Length byte of a set *n* indicates the Length of the Value field of the set. The set UL of a set *n* defines the complete list of metadata items present in the set.

Each item is KLV encoded, comprising a 2 byte Local Tag , 2 Length bytes and the data in the Value field V. The Set UL of the set defines the meanings of the 2 byte Local Tags of the items of the set.

It will be appreciated that a set *n* may itself contain a plurality of KLV encoded sets of data. An item may contain a set of KLV encoded data.

The optional Index Table

The index table provides a means of rapidly locating specific data, e.g. video frame starts, in the File Body.

The index table is an optional metadata set associated with the Body, which can be used to locate individual frames of video essence and related audio and data essence. Index Files may be placed either at the start of the Body, at the end of the Body or optionally distributed throughout the Body.

An index table can be created 'on the fly' during File creation from a stream input and is notionally placed at the end of the Body on recording. In practice, its placement in a server File system may be anywhere for storage convenience. During transfer of a complete File, the Index table is placed at the start of the Body.

Index tables are not necessary for Body container specifications which are defined with a constant number of bytes per frame, so their inclusion in the MXF File is optional. The Value of the Length of a frame which is constant may be stored in the preamble as an item of partition metadata.

File Body Essence Container $K_BL_BV_B$ Packet, Figure 3.

In accordance with the first example of the invention, a File Body essence container packet $K_BL_BV_B$ is provided in the File Body. The Value field V_B in this example contains an essence container of undefined Length. The Key field K_B has 16 bytes. The Length field L_B has a hexadecimal Value 80h. The Value 80h is defined in the SMPTE standard SMPTE 336M as defining an undefined Length of the Value field V_B.

The Key field K_B of the essence contianer packet $K_BL_BV_B$ may be as shown in Table 1 as follows:-

Byte No.	Description	Value (hex)
1	Object Identifier	06h
2	Label size	0Eh
3	ISO designator	2Bh
4	SMPTE designator	34h
5	Registry category: Wrappers/Containers	03h
6	Registry designator: Simple Wrappers/Containers	01h
7	Structure Designator: MXF Body Definitions	02h
8	Version Number	01h
9	Body Container Classification	xxh
10	Body Essence Classification	yyh
11	MXF Body Status	01h/02h
12~16	Zero fill	00h

Table 1: Label Value for the MXF Body Container Universal Label

The Key of the File Body $K_BL_BV_B$ is used to indicate :

1) the kind of Body container (e.g. a CP container);
2) the kind of essence type in the container (e.g. MPEG video) and
3) that the essence container V_B is included in the essence container of the File Body.

In the first example of the invention, the Byte No 11 has the Value 01h to indicate the essence container V_B is embedded in the MXF File as shown in Figure 1. A Value 02h of the byte 11 indicates that an essence container is not embedded in the File Body as will be described below.

It will be appreciated that the Table 1 is only an example and the definitions and Values of each byte of the Key will be specifically defined for each application which uses this invention. Also a different Value of the Key (in this example, byte 11) could be used to indicate that the essence container V_B is embedded. Further more, a Value other than 80h could be used to indicate an undefined Length of the Value field.

The essence container V_B, Figure 1 and 3

As shown in Figure 3, the essence container V_B comprises a sequence of KLV encoded packets. The Length field (80h) L_B is followed immediately by the Key of the first KLV packet of the essence container V_B. The amount of data in the essence

container V_B is unlimited and bounded only by the File Header and File Footer. The data in the essence container V_B could be for example many Gigabytes.

The present invention is relevant to any content of the essence container V_B. The contents in practice depend on the video essence, and/or audio essence and/or data essence contained in it and the manner in which it is encoded. For example, video and audio may be compressed or uncompressed. Several different forms of compression are possible. In the case of an SDTI-CP (Content Package), which contains different essence types, each type is separately KLV encoded. Each container type has a unique and registered Universal Label (UL) as a Key in the KLV coding. Each essence type within a container may have a unique Key.

The byte no. 9 of the Key shown in table 1 may be used to identify the type of essence container V_B and byte 10 may be used to identify the essence contained in the container.

The File Body is formatted as a stream of data packets and may use commonly available essence containers for example:

CP packets based on items used in SDTI-CP (see SMPTE 326M);

DV-DIF packets for video systems based on the DV compression system (see ISO/IEC 61834);

PS packets based on MPEG Program Stream Packets (see ISO/IEC 13818 Part 1);

Other packets based on uncompressed video and audio; and

Other packets based on other compressed video and audio systems.

The File Footer, Figure 1

The MXF File is terminated by a File Footer. The Footer contains a **Post-amble.**

The Post amble comprises a KLV encoded post-amble data set The post-amble data set comprises a SMPTE set label of 16 bytes, and a Length field (shown as 4 bytes in this example). In this example, the Value field comprises 84 to 384 bytes of partition metadata which defines aspects of data storage metadata used by the File for *use* in a sector based storage medium. An example of this metadata is an item which defines the sector size used in the storage medium and indicates that Key parts of the

File start at the beginning of a sector (e.g. the start of the essence container). This alignment to a sector boundary can improve access speed and allow editing of component parts of the File

The optional index table may be as described above.

MXF Header Metadata Repetition

The Header Metadata may be repetitively distributed through the File Body. The Header Metadata in the File Body is additional to the Header Metadata in the preamble. This has the advantages that:

if a File is interrupted during a transfer recovery of metadata is possible; and

an MXF File allows random access to data within the File Body. The repetitive metadata allows easier and quicker access to metadata relating to randomly accessed data because it avoids the need to scroll to the beginning or end of the File.

Repetition of Header metadata in the File Body is optional.

The foregoing description sets out some of the basic features of MXF Files. There are other features but they are not relevant to the present invention.

Second Example of an MXF File, Figure 4

The second example of the MXF File comprises a File Header as described with reference to Figures 1and 2, a File Footer as described with reference to Figure 1 and a File Body. The File Body is encoded having a Key field K_U, a Length field L_U and a Value field V_U. It will be noted that in contrast to the first example, the Value field V_U of the MXF File has no essence container embedded in it. Instead the Value field V_U contains a locator referring to the location of essence container (which would otherwise be contained in the Value field V_B) stored in a storage device. In this second example of the MXF File the locator is a Unique Value (UV) which may either be a URL (Uniform Resource Locator) or a Unique Identifier (UID).

A URL points to the location of the essence container. The URL as used in web browsers is one example of a locator which may be used. A different locator could be used, for example a Filename Value in a storage device or a VTR address. The locator may be the identity of a VTR or tape on a VTR together with start and stop time codes.

A UID provides a unique identification of the essence container. This UID may be used by a database to provide the location of the essence container. The advantage of a UID over a URL is that the essence container can be moved requiring only that the database mapping of the UID to the URL be changed. All references are then updated automatically. This technique requires a central database to resolve the UID to URL mapping but prevents the common 'broken link' problem found in web applications.

The essence container is not embedded in the MXF File, but instead is stored in a storage device at a location defined by the locator. The storage device may be a tape recorder in which case the essence container may comprise a sequence of KLV packets as shown in Figure 3.

The Key field K_U of the Second Example.

The Key field K_U is as described above and preferably has the form shown in Table 1. The byte 11 however has a different Value e.g. 02h Value which indicates that there is no essence container in the Value field V_U and that the essence container is in a location pointed to by the URL (or indirectly by a UID) in the Value field V_U.

The Length Field L_U of the Second Example.

The Length field L_U has a Value indicating the actual Length of the Value field V_U.

Index Table of the Second Example.

In the second example of the invention, the File Body may include an index table as described above. The index table is used to locate individual frames in the Body where that Body is stored in a remote location. In this case, the Key of the File Body contains a direct or indirect pointer to the location of the Body and the index table can be used to indicate the location of a frame without direct access to the remote location.

This second example allows the fast transfer of File data because the essence container, which may be many gigabytes, is not embedded in the MXF File.

Third Example, Figure 5.

This third example is a modification of the second example. Like the second example, this third example of the MXF File comprises a File Header as described

with reference to Figures 1 and 2 (subject to a modification described below), a File Footer as described with reference to Figure 1 and a File Body. The File Body is encoded having a Key field K_U, a Length field L_U and a Value field V_U. It will be noted that in contrast to the first example, the Value field V_U of the MXF File has no essence container embedded in it. Furthermore unlike the second example, the Value field V_U contains not a direct essence container locator which itself identifies the storage location of the essence container, but instead a an indirect locator which (i) indicates that the essence container locator is in the Header Metadata and (ii) indicates where the essence container locator is in the Header Metadata. It will be recalled that the Header Metadata may comprise a complex multi–layer arrangement of sets of items. By including the indirect pointer in the Value field V_U the existence of the essence container locator is simply indicated at a high level in the data structure allowing a simple parser to determine the existence of the locator.

The Key field K_U indicates that the File Body contains an indirect locator instead of the essence container or the essence container locator. The indirect locator points to the label of the metadata set in the Header Metadata which contains one or more essence container locators. The metadata set may contain UID Values which can be used to find one or more essence container locators. The Length field L_U indicates the actual amount of data in the Value field V_U.

Fourth Example Figure 6.

Referring to Figure 6, the MXF File comprises a Header as described with reference to Figures 1 and 2, a Footer as described with reference to Figure 2, and a Body comprising two components. The first component comprises a Key field K_U, a Length field L_U and a Value field V_U. The second component comprises a Key field K_B, a Length field L_B and a Value field V_B.

The Value field of one component (in Figure 6 the first component) contains a URL (or a UID) to the location of the Body container. The Value field of the other component (second component in Figure 6) is a Body container which contains essence.

In a first version of this format, the first component (K_U, L_U and V_U) has a Key which identifies the packet Value as containing a locator (either as a direct or indirect pointer or as a UID) which can be used to the locate the Body container which

contains a full resolution version of the essence. The Length field (L_U) has a Value which defines the Length of the locator in the Value field (V_U). The first component may be followed by a KLV Filler item which can be used to pad the File so that the second component lies at a defined position. The second component (K_B, L_B and V_B) has a Key (K_B) which identifies the packet Value as a Body container which contains a low resolution proxy representation of the Body container located by the first component. The Length field (L_B) has a pre-determined Value of 80h to indicate an undefined Length and the Value is the Body container with a low resolution proxy essence. This version of the format allows a full resolution Body to be stored in a remote location while still having a low resolution proxy video directly viewable from the File.

In a second version of this format, the first component (K_U, L_U and V_U) has a Key which identifies the packet Value as containing a direct or indirect locator to the location of the Body container which contains a low resolution proxy version of the essence. The Length field (L_U) has a Value which defines the Length of the locator in the Value (V_U). The second component (K_B, L_B and V_B) has a Key (K_B) which identifies the packet Value as a Body container which contains the full resolution representation of the Body container located by the first component. The Length field (L_B) has a pre-determined Value of 80h to indicate an undefined Length and the Value is the Body container with the full resolution essence. This version of the format allows a remotely stored File to point to the location where a low resolution proxy File can be found.

Referring to Table 1 above, this fourth example uses first and second Body components each having KLV packets. In the first Body packet, byte 11 of the Key field K_U indicates that the Value is a locator which directly or indirectly identifies the location of the Body container which contains either full resolution essence or a low resolution proxy essence through a URL or a UID. In the second Body packet, byte 11 of the Key field K_B indicates that the Value is a Body container which contains either full resolution essence or low resolution proxy.

Referring to figure 6, both the Body components include optional index tables. A first index table which immediately precedes the first $K_UL_UV_U$ packet containing the locator provides indexing for the Body container stored in the remote location. A

second index table which immediately precedes the second $K_BL_BV_B$ packet containing the embedded essence container provides indexing for the embedded essence. This second index table may also be present at the end of the second KLV packet or divided into partial index tables distributed throughout the embedded essence container.

The low resolution proxy essence may be contained in KLV items as shown in Figure 6 for example.

Whilst the examples refer to video, the essence may be any one or more of video, audio and data essence. For example, any File Body may contain audio where the stored essence is for example audio.

Any File Body may contain essence which is watermarked, encrypted or otherwise encoded to protect it against unauthorised use or to enable the owner to detect unauthorised use.

The KLV packet forming the File Body provides a consistent form for an MXF File which has essence embedded in the File Body and for an MXF Files which there is no embedded essence but which includes a locator directly or indirectly indicating the location of the stored essence container. Thus an MXF File structure can be used for both embedded essence and referenced essence.

Illustrative System, Figure 7

The system of Figure 7 includes a network 18 which may be any suitable network, e.g. Ethernet. A material source 2 provides material. The source may be a camera producing video essence for example. The source may alternatively be an audio source or a data source. The source may originate material or may reproduce recorded material. The following example assumes the source is a video camera. The video from the camera 2 is fed to an interface 6 which creates a container containing the video. The container may be any known container, but in this example it is an SDTI-CP container as known from SMPTE 326M. A source of metadata 4 is provided. The metadata relates to the material from the source 2. An MXF File creator or encoder 80 receives the metadata and the SDTI-CP input and creates a File structure as shown in Figures 1, 2 and 3 and described above. The CP container derived from a SDTI-CP input forms the essence container in the Value field V_B of the File Body. The

metadata forms the Header Metadata of the File Header. The Key K_B of the File Body includes the byte 11, the Value (01h) of which indicates that the structure includes the essence container in the Value field V_B.

A workstation 10 is linked to a search engine 12. The search engine is arranged to search for material stored in one or more stores 14 , 16. In general there are S stores. Material is stored in locations identified by locators, for example URLs or UIDs. In this example the workstation 10 together with the search engine 12 searches for material stored in the store(s) 14, 16. The stores 14, 16 may store metadata with the material and/or the workstation 10 may be used to generate metadata relating to material which is found. The workstation 10 may be used to generate the low resolution essence of the second version of the fourth example shown in Figure 6.

The URL (or UID) of material which is found together with the relevant metadata is fed from the workstation to an MXF encoder 81. The encoder 81 creates a File structure as shown in Figure 4, Figure 5 or Figure 6. The metadata forms the Header Metadata of the File Header and the URL (or UID) is included in the File Header preferably in the Header Metadata.

Referring to Figure 4, the encoder 81 produces the Key K_u of the File Body, which Key includes the byte 11, the Value (02h) of which indicates that the Value field V_u does not include the essence container, but instead includes the locator indicating directly or indirectly where the essence container is stored.

Referring to Figure 5, the encoder 81 produces the Key K_U of the File Body, which Key includes the byte 11, the Value of which indicates that the Value field V_U does not include the essence container, but instead includes the indirect locator indicating (i) that the Header Metadata includes a locator indicating where the essence container is stored; and (ii) indicating where the essence container locator is in the Header Metadata. The encoder 81 encodes the pointer in the Value field V_U as shown in Figure 5 and encodes the locator in a KLV encoded item in the Header Metadata.

Referring to Figure 6, the encoder 81 produces, for the first version of the fourth example, the said first component of the File Body having a Key K_U which identifies the packet as containing a locator indirectly or directly indicating the location of full

S6. Step S6 the MXF decoder 20 determines from the Key K_U of the first component of the File Body that the Value field of the first component contains a locator.. Step S6 also determines from the Key K_B of the second component whether the Value field V_B of the second component contains proxy essence or full resolution essence.

S8. The interface 22 extracts the essence from its container. The container may be a CP container for example, and the container is then mapped to an SDTI interface.

S10. The extracted essence container may stored in the store 24 which may be a VTR if the essence is video especially if it is full resolution essence and/or it may be displayed on the display 32 for browsing especially if it is the low resolution proxy essence.

S12. The locator is extracted from the Value field V_U of the first component by the decoder 20 and fed to the workstation 26 with the metadata.

S14. The locator is used by the workstation to retrieve the stored essence which may be low resolution proxy essence or full resolution essence.

CLAIMS

1. A data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with data in a File Body;

at least one of a first KLV packet forming a first component of a File Body and a second KLV packet forming a second component of the File Body;

the first component , if present, having a Value field containing location data directly or indirectly indicating the location of first essence which is not included in the File Body, and a Key field the Value of which indicates that the Value field of the first component contains the said location data;

the second component, if present, having a Value field containing an essence container for containing second essence, and a Key field the Value of which identifies the second essence;

one or both of the first and second essence being associated with the said metadata; and

a File Footer;

wherein the Key field of the first components has the same form as the Key field of the second component, the components being distinguished by distinguishing codes in the Key fields.

2. A data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with data in a File Body;

a KLV packet forming at least part of the File Body and having a Value field containing an essence container, a Key field including a predetermined code indicating

the Value field contains an essence container and, a Length field containing a predetermined code independent of the amount of data in the Value field; and

a File Footer.

3. A structure according to claim 2, wherein the said code in the Key field has a hexadecimal Value 01h.

4. A structure according to any preceding claim, wherein the said code in the Length field has a hexadecimal Value 80h.

5. A data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, the said KLV packets being organised into:

a File Header including a data space for metadata associated with essence;

a KLV packet forming at least part of the File Body having a Key field, a Value field and a Length field indicating the amount of data in the Value field; and

a File Footer;

wherein the Value field of the said a KLV packet forming at least part of the File Body contains location data directly or indirectly indicating the location of the essence container which essence is associated with the said metadata and which is not contained in the Value field of the File Body and the Key field of the said KLV packet forming at least part of the File Body has a predetermined code indicating that the Value field contains the said location data.

6. A structure according to claim 5 wherein the Key field of the File Body indicates that the Value field contains location data.

7. A structure according to claim 5 or 6, wherein the Key field of the File Body has a byte of predetermined Value which indicates that the essence container is not contained in the Value field of the File Body.

8. A structure according to claim 7, wherein the said byte has a hexadecimal Value 02h.

9. A structure according to claim 5, 6, 7 or 8, wherein the said location data is a Uniform Resource Locator.

10. A structure according to claim 5, 6, 7 or 8, wherein the said location data is a Unique Identifier usable to determine a Uniform Resource Locator for accessing the said essence container.

11. A structure according to claim 5, 6 or 7, wherein the said location data in the Value field indicates that a locator indicating the location of the essence container is in the said data space for metadata.

12. A structure according to claim 11 wherein the said location data indicates the location of the locator in the said data space for metadata.

13. A structure according to claim 11 or 12, wherein the locator is a Uniform Resource Locator or a Unique Identifier.

14. A structure according to any one of claims 5 to 13, wherein the Value field of the File Body contains other data.

15. A structure according to claim 14, wherein the said other data relates to the essence.

16. A structure according to claim 15, wherein the said other data is derived from the essence.

17. A structure according to claim 16, wherein the said other data comprises images.

28. A data structure according to any one of claims 5 to 17, wherein the file body includes a filler KLV packet for containing filler data so that the file body has a predetermined length.

29. A data structure according to any preceding claim wherein the said file header includes a filler KLV packet for containing filler data so that the file header has a predetermined length.

30. A structure according to any preceding claim, which is an MXF File structure.

31. A data structure substantially as hereinbefore described with reference to: Figure 1 optionally together with Figure 2 and/or 3; or Figure 4; or Figure 5 of the accompanying drawings.

32. A data structure substantially as hereinbefore described with reference to Figure 6 of the accompanying drawings.

33. A signal including a data structure according to any preceding claim.

34. A method of creating a data structure, comprising the steps of:
receiving digital data representing essence;
receiving metadata relating to the said essence; and
forming a data structure as specified in any one of claims 1 to 32.

35. A method of creating a data structure, comprising the steps of:
receiving digital data representing essence;
receiving metadata relating to the said essence; and
receiving location data relating to the location of further essence; and
forming a data structure comprising KLV packets each having Key, Length and Value fields, wherein a Key field denotes the type of data in a packet, a

Value field contains the data of the packet and a Length field denotes the amount of data in the Value field, ;

the method comprising organising the said KLV into:

a File Header including a data space for metadata associated with data in a File Body;

at least one of a first KLV packet forming a first component of a File Body and a second KLV packet forming a second component of the File Body;

the first component if present, having a Value field containing location data directly or indirectly indicating the location of first essence which is not included in the File Body, and a Key field the Value of which indicates that the Value field of the first component contains the said location data;

the second component, if present, having a Value field containing an essence container for containing second essence, and a Key field the Value of which identifies the second essence;

one or both of the first and second essence being associated with the said metadata; and

a File Footer;

wherein the Key field of the first components has the same form as the Key field of the second component, the components being distinguished by distinguishing codes in the Key fields.

36. A method according to claim 35, wherein the said first mentioned essence and the said further essence contain the same data but with different resolutions.

37. A method according to claim 36 or 36 comprising the further step of providing, in the first component of the file body, a filler KLV packet containing filler data so that the first component has a predetermined length.

38. A method according to claim 36 or 36 comprising the further step of providing, in the Header, a filler KLV packet containing filler data so that the Header has a predetermined length.

39. A method of creating a data structure substantially as hereinbefore described with reference to: Figure 1 optionally together with Figure 2 and/or 3; or Figure 4; or Figure 5; or Figure 6 of the accompanying drawings.

40. A method of creating a data structure substantially as hereinbefore described with reference to Figure 7 of the accompanying drawings.

41. A method of decoding a data structure, comprising the steps of:

receiving a File structure which may be as defined in any one of claims 1 to 17;

parsing the structure;

determining from a KLV packet having the predetermined Key field and a Length field whether the Value field of the KLV packet contains the essence container or contains location data indicating directly or indirectly the location of the essence container;

if the said Value field contains essence container extracting the essence container from the Value field and if the Value field contains the said location data extracting the location data therefrom.

42. A method according to claim 41, comprising the further step of retrieving from a store the essence container identified by the said location data.

43. A method according to claim 42, wherein the said location data in the Value field indicates that a locator is in the said data space for metadata and comprising the step of extracting the locator from the said data space for metadata.

44. A method according to claim 43, comprising the further step of retrieving from a store the essence container identified by the said locator.

45. A method of decoding a data structure, comprising the steps of:

receiving a File structure which may be as defined in any one of claims 18 to 28;

parsing the structure;

extracting the said location data from the first component if present;

extracting the said second essence container from the second container if present; and

accessing the said first essence from the location indicated by the said location data of the first component.

46. Apparatus arranged to create a File structure, as specified in any one of claims 1 to 32.

47. Apparatus arranged to carry out a method according to any one of claims 30 to 45.

48. A system for decoding a File structure, the processor being arranged to:

receive a File structure which may be as defined in any one of claims 1 to 28;

parse the structure;

determine from a KLV packet having the predetermined Key field and a Length field whether the Value field of the KLV packet contains the essence container or contains location data indicating the location of essence container; and

if the said Value field contains the essence container extract the essence container from the Value field and if the Value field contains the said location data extract the location data therefrom.

49. A system according to claim 48, including a store for storing an essence container and arranged to retrieve from the store the essence container identified by the said location data.

50. A system according to claim 49, wherein the said location data in the Value field is a pointer indicating that a locator is in the said data space for metadata and the system is arranged to extract the locator from the said data space for metadata.

51. A system according to claim 50, arranged to retrieve from the store the essence container identified by the said locator.

52. A system for decoding a File structure, the system being arranged to:

receive a File structure which may be as defined in any one of claims 18 to 28;

parse the structure;

extract the said location data from the first component;

extract the said second essence container from the second container; and

access the said first essence from the location indicated by the said location data of the first component.

53. A data processing system substantially as herein before described with reference to Figures 6 and 7 together with: Figure 1 optionally together with Figure 2 and/or 3; or Figure 4; or Figure 6; of the accompanying drawings.

54. A computer program product arranged to carry out the method of any one of claims 34 to 45 when run on a data processing system.